

On the number of MPLS LSP using Multicast Tree Aggregation

Joanna Moulrierac*, Alexandre Guitton† and Miklós Molnár*

* INRIA, University of Rennes I

† Birkbeck College, University of London

Phone number: +33 2.99.84.71.94

Email: joanna.moulrierac@irisa.fr, alexandre@dcs.bbk.ac.uk and
miklos.molnar@irisa.fr

Abstract—Multicast tree aggregation is an efficient proposition that can solve the multicast forwarding state scalability problem. Existing works on tree aggregation have focused on developing and simulating protocols that build trees dynamically. However, the underlying problem of the impact of the tree construction algorithm on the performance of the protocols remains untouched. In this paper, we propose a study on the number of trees that need to be configured in a domain depending on the tree construction algorithm. We ran extensive simulations on several real domains and with different tree construction algorithms. Our results show that for a given set of multicast groups, even when this set includes all the possible groups, the number of trees needed to be configured is small. This allows a network administrator to configure off-line all these trees in order to maintain a stable set of trees and to have knowledge of the routes used by the multicast packets. Knowing the set of all the possible trees is also useful to determine the best of its subset to configure and to give an upper bound of the number of different trees.

Keywords—Multicasting, tree aggregation, MPLS, control and management of network.

I. INTRODUCTION

The research in multicast has highly interested the researchers during the last decade but unfortunately, multicast is not well deployed yet over internet. However, as the number of multimedia applications including video on demand, television on internet and on line video games increases tremendously, multicast is becoming a necessity for network operators, in order to spare the network resources.

The multicast tree aggregation is an efficient proposition that allows the deployment of group communications. The main goal of tree aggregation is to keep only a small number of multicast forwarding states in the routers in order to spare the memory of the routers, to speed up the IP look-ups and to reduce the amount of control messages needed to maintain the forwarding states. Indeed, in multicast, aggregation of routing entries is more difficult than in unicast as the multicast IP addresses do not reflect the location of the members.

To reduce the number of entries, the multicast tree aggregation technique allows several multicast groups to share the same delivery tree. In other words, the packets for the groups can be multiplexed at entrance of the domain onto the same tree. This can be achieved using IP encapsulation or MPLS labels. In this case, the MPLS labels can be distributed among the routers using LDP.

Because of these two different ways of deploying multicast tree aggregation, we will talk in this paper indifferently about trees and about MPLS LSP.

Existing works on tree aggregation have focused on developing and simulating protocols that build trees dynamically. In this paper, our key concern is to study the impact of different tree construction algorithms on the performance of the protocol, which is still a research issue. The performance of a tree aggregation protocol is defined by the number of trees to be maintained and by the resource usage for each group. We first show that the way trees are built leads to major scalability issues. Then, we study several commonly used tree construction algorithms, along with the number of trees to be maintained in order to cover a set of multicast groups. We show with our simulations that it is possible to build a set of trees covering all the possible multicast groups in a given domain as this number is usually very small and few forwarding entries are needed in routers. This allows a network administrator to configure this stable set of trees in order to plan the routing for the future multicast groups, to be aware of the routes used by the multicast packets and to maintain a stable set of forwarding entries. Moreover, minimizing the number of MPLS LSP in a domain is of paramount importance for scalability issues.

The rest of the paper is organized as follows. Section II presents the main protocols achieving tree aggregation. In Section III, we present the main idea of our proposition and give an algorithm that computes the number of MPLS LSP required, *i.e.*, the number of multicast trees. The results of the simulations are detailed in Section IV. Finally, the perspectives of our work are given in Section V.

II. MULTICAST TREE AGGREGATION

Tree aggregation reduces the number of multicast forwarding states and the tree maintenance overhead. To achieve this reduction, several multicast groups share the same delivery tree within a domain. Consequently, less trees are built and less forwarding entries are stored than with traditional multicast.

A. Aggregation protocols

The protocol AM (Aggregated Multicast) was proposed in [1], [2]. In AM, a centralized entity called the tree manager

is in charge of matching groups to labels and informing the border routers of this matching.

Several recent protocols have addressed the scalability problems of AM. For instance, STA [3] performs faster aggregations than AM by sorting the trees in the multicast tree set, addressing the time-scalability issue. In BEAM [4], the task of the tree manager is distributed among several routers, called core routers. The protocol AMBTS [5] splits the native tree in sub-trees before aggregating, while the protocol TALD [6] performs tree aggregation in large domains by splitting the network in several sub-domains and by managing the sub-domains independently. AMBTS and TALD can be applied in large domains, where other approaches fail. The advantage of TALD over AMBTS is that no entity has the total knowledge of the domain.

In parallel, new protocols have been developed with additional constraints. The paper [7] studies ASSM protocols designed for source-specific groups. AQoS [8] and Q-STA [9] perform tree aggregation under bandwidth constraints. In these two protocols, the links have limited bandwidth capacities and groups have bandwidth requirements depending on the multimedia application. Indeed, previous approaches assume links have infinite bandwidth capacity and therefore focus on the aggregation ratio. The protocol TOMA [10] deals with overlay multicast. Finally, the protocol AMFM [11] proposes a fault tolerant architecture.

In all these protocols, the tree construction algorithm is either unspecified or arbitrary. We believe that understanding how the tree construction algorithm impacts tree aggregation protocols can lead to further enhancement of the existing mechanisms, or to protocols based on new concepts.

B. On the number of trees

Our work is based on the frameworks proposed in [12] and in [13]. Among the research papers on tree aggregation, the closest to our work is [14], which also considers the number of trees in a domain. However, our work tackles the problem from another angle. First, we consider that the branching routers of the trees can lead to multicast members. Indeed, in real networks, the non-leaf routers used by a tree can be multicast capable and can be attached to members. Second, we propose an analysis of the upper-bound of the number of trees needed to span the set of all multicast groups considering different tree algorithm construction. Some scalability problems arise with this approach when the number of routers in the network is large, as the set of all the multicast groups can not be computed. However these problems can be solved (although not optimally) by using a similar approach as TALD, *i.e.*, by splitting the domains into sub-domains and computing the set of trees into each sub-domain separately.

As we have said earlier, the previous works do not focus on the impact of the tree construction algorithms on the number of MPLS LSP or trees to be maintained. However, we believe that it is one of the most important parameter of tree aggregation protocols. That is why we are interested in answering the following questions:

- What is the impact of the tree construction algorithm on the number of different trees?

- How does the number of different trees evolves as a fraction of the bandwidth is allowed to be wasted?

III. ALGORITHM FOR CONFIGURING A SET OF TREES

In this section, we present an efficient way to reduce the number of trees and then we present the algorithm that computes a set of multicast trees for a given domain.

First of all, when we say that a router is a member of a group, it means that this router leads to group members. The real members of the group can be directly attached to the router or can be connected to it through several domains. In this way, we will define by group the set of the routers involved in the group. With this definition, two groups can be identical even if they do not have the same multicast address.

Second, since the main goal of the tree aggregation is to reduce the number of trees, we will consider shared tree to be built as usually made by tree aggregation protocols. However, our approach still holds when considering source-based trees. It can be noticed that for source-based trees, more trees are configured as all (source,group) tuples are considered whereas in case of shared trees, only the groups are considered. In the literature, shared trees are build with CBT [15] or with PIM-SM [16]. The trees are rooted at a core router or at a *rendez-vous point* and cover the members of the groups using shortest-path computed with Dijkstra algorithm. As the determination of this source is still a research issue in the literature, we will consider in this paper three different ways to chose this source for the groups and this is described later in Section IV. Shared trees can also be build with the objective of minimizing the cost and then approximate Steiner trees can be used [17]. Then, even if such trees are not used in real networks, it gives a lower bound of the bandwidth to be used for a set of groups.

A. On reducing the number of trees

Previous works on tree aggregation were based on the following intuitive remark: two different multicast groups can be the same when projected on a domain. In other words, if we consider the routers that have members attached to them, it is easy to see that different members can be attached to the same set of edge routers. If n is the number of routers of the domain, the upper bound on the number of trees is only 2^n , instead of having as an upper bound the whole space of multicast addresses.

However, we can go a step further. Let us consider the example shown on Fig. 1. A tree, rooted at A and covering the routers $\{A B C D E F\}$ is represented on the figure. This tree is the native tree for four different groups. Indeed, it can cover the group with members attached to routers $\{A C D F\}$ but also the group $\{A C D E F\}$, the group $\{A B C D F\}$ and the group $\{A B C D E F\}$. All these aggregations can be done without bandwidth wasted as the tree traverses the two branching routers B and E in order to reach the leaves C and F .

Therefore, the number of different trees is lower than the previous bound 2^n . We only noticed that packets for the groups traverse several routers that can be multicast capable before reaching the leaves.

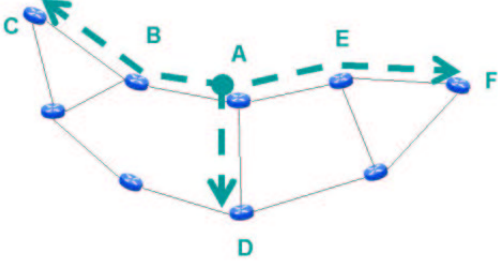


Fig. 1. The tree represented on the figure covers different multicast groups.

Note that in the previous description, no bandwidth is allowed to be wasted. To further reduce the number of different trees in a domain, it is realistic to consider that the network administrators allow a given percentage of the multicast bandwidth to be wasted in order to ease the aggregation process. Let us consider once more the example shown on Fig. 1. A group g having members attached to routers $\{B F D\}$ can share the same tree as the one depicted, but some bandwidth will be wasted when packets for the group g reach router C unnecessarily. If the bandwidth used by the group g is small compared to the bandwidth available on the link (B, C) , it could be better to have only one tree instead of two.

B. Description of the Algorithm

Algorithm 1 computes the set \mathcal{T} of different trees given the bandwidth bound bw . No more than bw percent of bandwidth are allowed to be wasted. Every possible group is considered in turn. For each group $g_i \in \mathcal{G}$, the algorithm computes a native tree t_i . If there is already a tree in \mathcal{T} that covers the same edge routers as t_i and that does not waste too much bandwidth, then there is no need to add t_i in the set \mathcal{T} .

Input: a domain with n routers, a tree construction algorithm \mathcal{A} , a bandwidth threshold bw
Output: a set of trees \mathcal{T} covering all the 2^n groups
 $\mathcal{T} \leftarrow \emptyset$
for $i = 1$ to 2^n **do**
 g_i is the i -th group of the 2^n possible groups of \mathcal{G}
 compute a native tree t_i covering g_i with algorithm \mathcal{A}
 if there is a tree $t \in \mathcal{T}$ that covers the routers covered by t_i **and if** $cost(t) \leq (cost(t_i) + cost(t_i) * bw)$ **then**
 t_i is ignored
 else
 add t_i to \mathcal{T}
return \mathcal{T}

Algorithm 1: Computing the set of different trees

The algorithm running-time scales rather badly with the number of edge routers of the domain. However, previous works [6] have shown that a large domain can be separated into several sub-domains having few edge routers, in such a way that the tree aggregation can be done separately in each sub-domain without suffering a significant drop of the

performance. Therefore, if the domain is too large, it can be divided into several sub-domains, and then the algorithm presented above can be applied in each of the sub-domain in order to scale better.

The number of trees computed by Algorithm 1 applies for a worst-case scenario where each group has an equal probability of being requested. Previous studies have shown that real groups show a high degree of correlation and then they can be simulated using some affinity model. Our algorithm still applies for this scenario, and in this case, the number of trees needed to be configured is smaller. Moreover, our algorithm can be applied for only a set of multicast groups to be covered. This can be useful when a network administrator has traces of real multicast groups and wants to configure only a set of trees covering these groups.

IV. SIMULATION RESULTS

We made simulations of Algorithm 1 on different real graphs: the VSNL network [18] (which is given by an internet mapping tool called Rocketfuel [19]), the Abilene network [20], the Nsfnet network [21] and the Geant network [22]. We ran the simulations for two bandwidth thresholds: when no bandwidth is wasted ($bw = 0\%$) and when 20% of bandwidth is allowed to be wasted. In order to evaluate the number of trees by a comparison of the tree algorithm construction, we consider different ways of choosing the core or the *rendez-vous point* for CBT or PIM-SM. Recall that these protocols build shortest path trees rooted at a given core. We also consider MST trees. The compared algorithms are:

- CBT-Fixed: a core router is chosen arbitrarily among the routers of the network initially and all the trees are rooted at this core.
- CBT-First: a core router is chosen among the members: first, the members are sorted according to a given order of the routers, and then the core is the first member.
- CBT-Rand: the trees are rooted at a core randomly chosen among the members.
- MST: a minimum spanning tree covering the routers-members is built in the metrical closure graph. This tree is a 2-approximate Steiner tree.

A. Number of trees to be configured

Table I shows the total number of different groups and the number of trees needed to cover these groups. Note that for a domain with n multicast capable routers, there are $(2^n - n - 1)$ different groups. Indeed, we take into account neither groups having one member nor groups having no member. The number of different trees is very low compared to the number of different groups. Therefore, it is possible to configure all the different trees in a domain.

When using CBT-Fixed, trees are very similar. Indeed, they are rooted at the same core and use frequently the same links (since shortest paths trees are built). This means that CBT-Fixed build few different trees. Oppositely, we can see that more trees are built with MST than with CBT-Fixed. Indeed, MST trees use less links and the link usage is distributed fairly in the network. Thus, the different trees built by MST are very

different with each other, and less aggregation can be found. However, with CBT-Fixed, the links around the core router experience an important stress, since they are shared by most of the trees.

With CBT-First, the trees have frequently the same core. Then, there are few different cores and the link usage of trees with CBT-First is fairer than with CBT-Fixed, but more trees are required. Finally, CBT-Rand chooses randomly the core for each group which implies that the trees are rooted at several different sources. The link usage is balanced but more trees are needed than with all others CBT algorithms. This shows that the choice of the core router has a great impact on the number of trees to be configured. With the MST algorithm, we can get an upper bound of the number of trees needed when the metric considered is the network usage minimization.

On the impact of the topology: The topology of the network has also a great impact on the number of trees. Indeed, although the Abilene and the VSNL networks share the same number of nodes, the number of trees built in both is different. VSNL contains fewer links and several routers have a degree of 1. Because of this small number of branching routers, more trees are configured for VSNL than for Abilene.

To explain this phenomenon, let us consider a network with n routers where $n - 1$ routers are leaves and are connected to the n -th router. This network is a star network. In this example, the number of trees is independent of the tree construction algorithm and is equal to $2^{n-1} - 1$ (the number of permutations of the $n - 1$ leaves minus the empty tree). In this example, the router linked to all others routers has to keep $2^{n-1} - 1$ forwarding entries. Another example to be considered is a path of n routers. In this network, there are 2 leaves and $n - 2$ routers with a degree of 2: the number of different trees is $\frac{n(n-1)}{2}$, which is much smaller than on the previous example. These two examples show the impact of the network topology on the number of trees.

B. Number of forwarding entries per router

The figures Fig. 2 and Fig. 3 plot the number of forwarding entries per router for the Geant network and for Abilene. We compare the minimum, the maximum and the mean number of forwarding entries to be stored per router. The figures show that when bandwidth wastage is allowed $bw = 20\%$, the number of forwarding entries is smaller than with $bw = 0\%$. The number of forwarding entries is strongly related to the number of trees and then the impact of tree construction algorithm for both metrics can be explained in the same way. What can be noticed, is that the number of forwarding entries is small enough to be realistic to envisage to stored them in routers.

C. Mean cost of trees per group

The figures Fig. 4 and Fig. 5 plot the mean cost of trees per group:

$$\frac{\sum_{g \in \mathcal{G}} \text{cost}(t(g))}{|\mathcal{G}|},$$

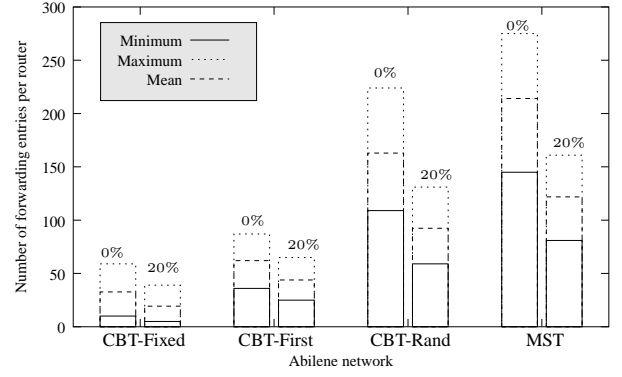


Fig. 2. Number of forwarding entries for Abilene network.

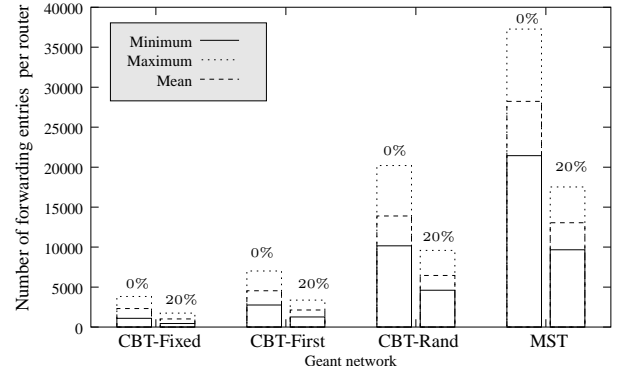


Fig. 3. Number of forwarding entries for Geant network.

where $t(g)$ is the tree of minimum cost in \mathcal{T} that covers g , $\text{cost}(t(g))$ is the cost of the tree $t(g)$ in terms of number of links and $|\mathcal{G}|$ is the number of different groups. This metric reflects the number of links loaded and then gives an overview of the bandwidth used per group. These two figures show the compromise between the number of trees and the usage of the network resources. Indeed, the less trees to be maintained and the more the cost of trees per group and then the more links are loaded. For Geant network, which has more than 260 000 different groups, a network administrator can configure less than 2 000 trees with CBT-Fixed and with $bw = 20\%$, but some bandwidth is wasted for each group, the trees have 12 links in average for each group. For a minimum bandwidth usage, MST with $bw = 0\%$ configures 75 000 trees and the trees have a little bit more than 9 links per group. It is interesting to notice that building MST trees with $bw = 20\%$ leads to approximately the same results as building CBT-Rand trees with $bw = 0\%$ for Geant network.

V. CONCLUSION AND FUTURE WORK

We proposed, in this paper, a study of the impact of tree construction on the number of trees to be maintained in a domain. During the simulations, we generated all the possible groups for a given domain and we computed the set of trees needed to cover all these groups while varying the tree construction algorithms. Our study showed that very few trees are needed to be configured in a domain and that it is possible for a network administrator to compute this stable set of trees

	bw	Number of \neq groups $ \mathcal{G} $	CBT-Fixed	CBT-First	CBT-Rand	MST
VSNL Network [18]	0%	2036	173	239	279	293
(11 routers, 12 links)	20%	2036	103	156	179	191
Abilene Network [20]	0%	2036	59	131	292	370
(11 routers, 14 links)	20%	2036	39	104	183	230
Nsfnet Network [21]	0%	16369	395	958	3299	4785
(14 routers, 21 links)	20%	16369	210	549	1740	2486
Geant Network [22]	0%	262125	3824	8222	24542	48942
(18 routers, 30 links)	20%	262125	1748	4126	11924	23449

TABLE I
TOTAL NUMBER OF GROUPS AND TOTAL NUMBER OF TREES

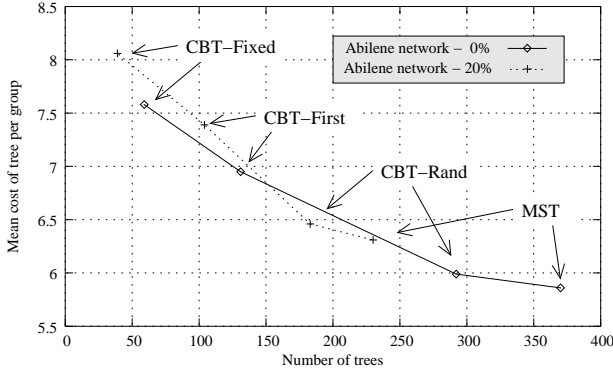


Fig. 4. Mean cost of trees per group for Abilene network [20].

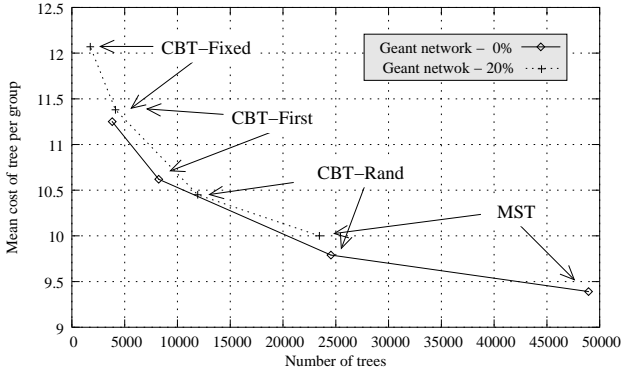


Fig. 5. Mean cost of trees per group for Geant network [22].

off-line. This allows to manage more efficiently the domain and to be aware of the configured trees. In this paper, we also studied in this paper the compromise between the number of trees configured and the mean cost of trees per group. When the network administrator has real traces of multicast groups, only a sub-set of all these trees can be computed. Indeed, we proposed an algorithm that gives an upper bound on the the number of trees needed depending on the tree construction algorithm. When these trees are configured, usual group-tree matching protocols can be achieved.

The number of trees is also dependent on the topology of the network. When the network topology implies that the number of trees is still too high, then several investigation has to be done to further reduce this number. We plan to design, in this case, a new tree construction algorithm that builds few trees.

REFERENCES

- [1] M. Gerla, A. Fei, J.-H. Cui, and M. Faloutsos, "Aggregated Multicast for Scalable QoS Multicast Provisioning," in *Tyrrhenian International Workshop on Digital Communications*, September 2001.
- [2] J.-H. Cui, J. Kim, D. Maggiorini, K. Boussetta, and M. Gerla, "Aggregated multicast — a comparative study," in *IFIP Networking*, ser. LNCS, no. 2345, May 2002, pp. 1032–1044.
- [3] A. Guitton and J. Moulhierac, "Scalable Tree Aggregation for Multicast," in *8th International Conference on Telecommunications (ConTEL)*, June 2005, best student paper award.
- [4] J.-H. Cui, L. Lao, D. Maggiorini, and M. Gerla, "BEAM: A Distributed Aggregated Multicast Protocol Using Bi-directional Trees," in *IEEE International Conference on Communications (ICC)*, May 2003.
- [5] Z.-F. Liu, W.-H. Dou, and Y.-J. Liu, "AMBTs: A Scheme of Aggregated Multicast Based on Tree Splitting," in *IFIP Networking*, ser. LNCS, no. 3042, May 2004, pp. 829–840.
- [6] J. Moulhierac, A. Guitton, and M. Molnár, "Multicast Tree Aggregation in Large Domains," in *IFIP Networking*, 2006.
- [7] J.-H. Cui, D. Maggiorini, J. Kim, K. Boussetta, and M. Gerla, "A Protocol to Improve the State Scalability of Source Specific Multicast," in *IEEE Globecom*, November 2002.
- [8] J.-H. Cui, J. Kim, A. Fei, M. Faloutsos, and M. Gerla, "Scalable QoS Multicast Provisioning in Diff-Serv-Supported MPLS Networks," in *IEEE Globecom*, November 2002.
- [9] J. Moulhierac and A. Guitton, "QoS Scalable Tree Aggregation," in *IFIP Networking*, ser. LNCS, no. 3462, May 2005, pp. 1405–1408.
- [10] L. Lao, J.-H. Cui, and M. Gerla, "TOMA: A Viable Solution for Large-Scale Multicast Service Support," in *IFIP Networking*, ser. LNCS, no. 3462, May 2005.
- [11] J.-H. Cui, M. Faloutsos, and M. Gerla, "An Architecture for Scalable, Efficient and Fast Fault-Tolerant Multicast Provisioning," *IEEE Network special issue on Protection, Restoration, and Disaster Recovery*, vol. 18, no. 2, pp. 26–34, March/April 2004.
- [12] J. Moulhierac, "On the number of multicast aggregated trees in a domain," in *2nd Student Workshop of IEEE Infocom*, April 2006.
- [13] J. Moulhierac and A. Guitton, "Distributed Multicast Tree Aggregation," INRIA, Research Report 5636, July 2005.
- [14] L. Lao, J.-H. Cui, and M. Gerla, "Tackling Group-Tree Matching in Large Scale Group Communications," UCLA CSD, Technical Report 040022, June 2004.
- [15] A. Ballardie, "Core Based Trees (CBT) Multicast Routing Architecture," IETF, RFC 2201, September 1997.
- [16] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. E. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei, "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification," IETF, RFC 2362, June 1998.
- [17] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for Steiner trees in graphs," *Acta Informatica*, vol. 15, pp. 141–145, 1981.
- [18] VSNL network, "http://www.cs.washington.edu/research/networking/rocketfuel/interactive/."
- [19] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with Rocketfuel," in *ACM SIGCOMM*, August 2002.
- [20] Abilene network, "http://abilene.internet2.edu."
- [21] Nsfnet network, "http://www.cybergeography.org/atlas/nsfnet.t1.gif."
- [22] Geant network, "http://www.cybergeography.org/atlas/geant.large.gif."